A RECURSIVE FEATURE ELIMINATING METHOD BASED ON A SUPPORT VECTOR MACHINE

BACKGROUND

[0001] A recursive feature eliminating method based on a support vector machine (SVM-RFE) is widely used in data intensive applications, such as disease genes selection, structured data mining, and unstructured data mining, etc. The SVM-RFE method may comprise: SVM training an input training data to classify the training data, wherein the training data may comprise a plurality of training samples corresponding to a group of features and class labels associated with each of the training samples; eliminating at least one feature with a minimum ranking criterion from the group of features; and repeating the aforementioned SVM training and eliminating until the group becomes empty. The SVM-RFE may be used to rank the features, for example, to rank the genes that may cause a disease. Rounds of SVM training and eliminating are independent with each other.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0003] Fig. 1 illustrates an embodiment of a computing system applying a SVM-RFE method.

[0004] Fig. 2 illustrates an embodiment of a SVM-RFE machine in the computing system of Fig. 1.

[0005] Fig. 3 illustrates an embodiment of a SVM-RFE method;

[0006] Fig. 4 illustrates an embodiment of a SVM training method involved in the SVM-RFE method of Fig. 3.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0007] The following description describes techniques for a recursive feature eliminating method based on a support vector machine. In the following description, numerous specific details such as logic implementations, pseudo-code, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the current invention. However, the invention may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0008] References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0009] Embodiments of the invention may be implemented in hardware, firmware, software, or any combination thereof. Embodiments of the invention may also be implemented as instructions stored on a machine-readable medium, that may be read and executed by one or more processors. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.) and others.

[0010] Fig. 1 shows a computing system for implementing a recursive feature eliminating method based on a support vector machine (SVM-RFE). A non-exhausive list of examples for the computing system may include distributed computing systems, supercomputers, computing clusters, mainframe computers, mini-computers, client-server systems, personal computers, workstations, servers, portable computers, laptop computers and other devices for transceiving and processing data.

[0011] In an embodiment, the computing system 1 may comprise one or more processors 10, memory 11, chipset 12, I/O device 13, BIOS firmware 14 and the like. The one or more processors 10 are communicatively coupled to various components (e.g., the memory 11) via one or more buses such as a processor bus as depicted in Fig. 1. The processors 10 may be implemented as an integrated circuit (IC) with one or more processing cores that may execute codes under a suitable architecture, for example, including Intel® Xeon™ MP architecture available from Intel Corporation of Santa Clara, California.

[0012] In an embodiment, the memory 12 may store codes to be executed by the processor 10. In an embodiment, the memory 12 may store training data 110, SVM-RFE 111 and operation system (OS) 112. A non-exhausive list of examples for the memory 102 may comprise one or a combination of the following semiconductor devices, such as synchronous dynamic random access memory (SDRAM) devices, RAMBUS dynamic random access memory (RDRAM) devices, double data rate (DDR) memory devices, static random access memory (SRAM), flash memory devices, and the like.

[0013] In an embodiment, the chipset 12 may provide one or more communicative path among the processor10, memory 11 and various components, such as the I/O device 13 and BIOS firmware 14. The chipset 12 may comprise a memory controller hub 120, an input/output controller hub 121 and a firmware hub 122.

[0014] In an embodiment, the memory controller hub 120 may provide a communication link to the processor bus that may connect with the processor 101 and to a suitable device such as the memory 11. The memory controller hub 120 may couple with the I/O controller hub 121, that may provide an interface to the I/O devices 13 or peripheral components (not shown in Fig. 1) for the computing system 1 such as a keyboard and a mouse. A non-exhaustive list of examples for the I/O devices 13 may comprise a network card, a storage device, a camera, a blue-tooth, an antenna, and the like. The I/O controller hub 121 may further provide communication link to a graphic controller and an audio controller (not shown in Fig. 1). The graphic controller may control the display of information on a display device and the audio controller may control the display of information on an audio device.

[0015] In an embodiment, the memory controller hub 120 may communicatively couple with a firmware hub 122 via the input/output controller hub 121. The firmware hub 122 may couple with the BIOS firmware 14 that may store routines that the computing device 100 executes during system startup in order to initialize the processors 10, chipset 12, and other components of the computing device 1. Moreover, the BIOS firmware 14 may comprise routines or drivers that the computing device 1 may execute to communicate with one or more components of the computing device 1.

[0016] In an embodiment, the training data 110 may be input from a suitable devices, such as the I/O component 13, or the BIOS firmware. Examples for the training data 110 may comprise data collected for a feature selection/ranking task, such as gene expression data from a plurality of human beings or other species, or text data from web or other sources. The data format may be structured data, such as a database or table, or unstructured data, such as matrix or vector. The SVM-RFE 111 may be implemented between the training data 110 and the operation system 112. In an embodiment, the operation system 112 may include, but not limited to, different versions of LINUX, Microsoft Windows™ Server 2003, and real time operating systems such as VxWorks™, etc. In an embodiment, the SVM-RFE 111 may implement operations of: SVM training the training data 110 that corresponds to a group of features; eliminating at least one feature from the group according to a predetermined ranking criterion; and repeating the SVM training and feature eliminating until the number of features in the group reaches a predetermined value, for example, until the group becomes empty, wherein the rounds of SVM training and eliminating dependent with each other. The SVM-RFE 111 may output a feature elimination history or a feature ranking list.

[0017] Other embodiments may implement other modifications or variations to the structure of the aforementioned computing system 1. For example, the SVM-RFE 111 may be implemented as an integrated circuit with various functional logics as depicted in Fig. 2. For another example, the memory 11 may further comprise a validation software (not show in Fig. 1) to validate the SVM-RFE classification by the SVM-RFE 111. More specifically, the validation software may determine whether a person has a disease by checking his/her gene expression with a gene ranking list output by the SVM-RFE 111.

[0018] An embodiment of the SVM-RFE 111 is shown in Fig. 2. As shown, the SVM-RFE 111 may comprise a decision logic 21, a SVM learning machine 22, a ranking criterion logic 23 and an eliminating logic 24.

[0019] In an embodiment, the training data 110 input to the SVM-RFE 111 may comprise a plurality of training samples $[x_1, x_2, \ldots, x_m]$ corresponding to a group of features, wherein m represents the number of training samples. The training data may further comprise class labels associated with each of the training samples $[y_1, y_2, \ldots, y_m]$. In an embodiment, each of the training samples represents a vector of n dimensions, wherein each dimension corresponds with each feature, and each of the class labels has a number of values. For example, if the training data is gene data collected from a plurality of persons, each of the training samples represents a pattern of n gene expression coefficients for one person, and each of the class labels has two values (i.e., [1, -1]) to represent two-class classification of its associated training sample, e.g., whether the person has a certain decease ($y_i = 1$) or not ($y_i = -1$).

[0020] In an embodiment, the decision logic 21 may determine whether the group is empty and output a feature ranking list or feature elimination history if so. However, if the group is not empty, the SVM learning machine 22 may train the training data by setting a normal to a hyperplane where the training data may be mapped to leave the largest possible margin on either side of the normal. The SVM learning machine 22 may comprise a linear SVM learning machine and non-linear SVM learning machine. In an embodiment for linear SVM learning machine, a normal may comprise a vector ($\vec{\omega}$) representing a linear combination of the training data. For non-linear SVM learning machine, a normal may comprise a vector ($\vec{\omega}$) representing a non-linear combination of the training data. Each component of the vector represents a weight for each feature in the group of features.

[0021] In an embodiment, the ranking criterion logic 23 may compute a predetermined ranking criterion for each feature based upon the weight vector $\vec{\omega}$. The eliminating logic 27 may eliminate at least one feature with a certain ranking criterion from the group of features, for example, the at least one feature with a minimum or maximum ranking criterion in the group of features. Then, the decision logic 21 may determine whether the group becomes empty. If not, then in another round of SVM training and feature eliminating, the SVM learning machine 22 will retrain the training data corresponding to the group of features without the eliminated ones, the ranking criterion logic 23 and eliminating logic 24 may compute the predetermined ranking criterion for each features in the group and eliminate at least one features with a minimum ranking criterion from the group of features. The SVM-RFE 111 may repeat the rounds of SVM training and feature eliminating as described above until the group becomes empty.

[0022] In an embodiment, the SVM learning machine 22 may comprise a kernel data logic 220, a buffer 221, a Lagrange multiplier logic 222 and a weight logic 223. In a first round of SVM training, the kernel data logic 22 may compute the kernel data based on the training data corresponding to the group of features and store the kernel data in the buffer 22 and then in each round of SVM training later, the kernel data logic 220 may retrieve a kernel data from the buffer 23, update the kernel data based on a part of the training data corresponding to the at least one feature that may be eliminated in a previous round and store the updated kernel data in the buffer in place of the old one.

[0023] In an embodiment, the Lagrange multiplier logic 222 may compute a Lagrange multiplier α_i for each of the training samples by utilizing the kernel data output from the kernel data logic 220 and the weight logic 224 may obtain a weight ω_k for each feature in the group of features, wherein i is an integer in a range of [1, the number of training samples], and k is an integer in a range of [1, the number of features].

[0024] Fig. 3 depicts an embodiment of a SVM-RFE method that may be implemented by the SVM-RFE 111.

[0025] As depicted, the SVM-RFE 111 may input the training data 110 in block 301. In an embodiment, the training data may comprise a plurality of training samples $[x_1, x_2, \ldots, x_m]$, wherein m represents the number of training samples. The training data may further comprise class labels associated with each of the training samples [$y_1, y_2, \ldots, y_m$]. Each of the training samples may represent a vector of n dimensions, wherein each dimension corresponds to each feature in a group of features (hereinafter, the group is labeled as group G), and each of class labels has a number of values to represent the class that its associated training sample belongs to.

[0026] In block 302, the decision logic 21 of SVM-RFE 111 may determine whether the number of features in the group G is zero (block 301). If the number of features in the group G is greater than zero, then the SVM learning machine 22 of SVM-RFE 111 may train the training data corresponding to the features in the group G, so as to obtain a vector ($\vec{\omega}$) for the training data (block 303). Each component of the weight vector represents a weight (e.g., weight (ω_k)) for a feature (e.g., the k^th feature) in the group G.

[0027] Then, the ranking criterion logic 23 may compute a ranking criterion for each feature in the group G based on its weight in block 304. In an embodiment, the ranking criterion is a square of the weight, e.g., $c_k = (\omega_k)^2$, wherein c_k represents the ranking criterion for the k^th feature. However, in other embodiments, the ranking criterion may be obtained in other ways.

[0028] In block 305, the eliminating logic 24 may eliminate at least one feature with a certain ranking criterion from the group G. In an embodiment, the at least one feature (e.g., the k^th feature) may correspond to the ranking criterion (e.g., $c_k = (\omega_k)^2$) that is the minimum in the group G. In another embodiment, the at least one feature may correspond to the ranking criterion that is the maximum in the group G. In other embodiments, the at least one feature may be eliminated in other ways.

[0029] In block 306, the eliminating logic 24 of the SVM-RFE 111 or other suitable logics may optionally update the training data by removing a part of the training data that corresponds to the eliminated features. In an embodiment that the input training data may comprise m training samples and m class labels associated with the training samples, and each of the training samples is a vector of n dimensions wherein each dimension corresponds to each feature of the group G, the updated training data may comprise m training samples and m class labels associated with the training samples, and each of the training samples is a vector of (n-p) dimensions wherein (n-p) represents the number of the features in the group G after p features may be eliminated in block 305.

[0030] In block 307, the eliminating logic 24 of the SVM-RFE 111 or other suitable logics may record the eliminating history, or record the feature ranking list based on the eliminating history. In an embodiment, the at least one features eliminated in block 305 may be listed as a least important feature in the feature ranking list. In another embodiment, the at least features may be listed as a most important feature in the feature ranking list.

[0031] Then, the decision logic 21 of the SVM-RFE 111 may continue to determine whether the number of features in the group G is zero in block 302. If not, the round of SVM training and feature eliminating as described with reference to blocks 303-307 may be repeated until the group G is determined to be empty, namely, the number of features therein is zero.

[0032] If the decision logic 21 determines the number of features in the group G is zero in block 302, then the decision logic 21 or other suitable logics of SVM-RFE 111 may output the eliminating history or the feature ranking list.

[0033] Fig. 4 depicts an embodiment of SVM training implemented by the SVM learning machine 22 in block 303 of Fig. 3. In the embodiment, blocks depicted in Fig. 4 may be implemented in each round of SVM training and feature elimination.

[0034] As depicted, the kernel data logic 220 of the SVM learning machine or other suitable logics may determine whether it is the first round of SVM training for the training data 110 (block 401). This determination may be accomplished by setting a count number. If it is the first round of SVM training, then the kernel data logic 220 may compute a kernel data based on the training data 110 in block 402. In an embodiment for linear SVM training, the kernel data may be computed by the following equations (1) and (2):

$$K^{round1} = \begin{bmatrix} k_{1,1}^{round1} \cdots\cdots k_{1,m}^{round1} \\ \cdots\cdots k_{i,j}^{round1} \cdots\cdots \\ k_{m,1}^{round1} \cdots\cdots k_{m,m}^{round1} \end{bmatrix} \tag{1}$$

$$k_{ij}^{round1} = x_i^T x_j = \sum_{k=1}^{n} x_{ik} x_{jk} \tag{2}$$

wherein, K^{round1} is the kernel data of a matrix with $(m \cdot m)$ components k_{ij}^{round1}, m represents the number of training samples, x_i^T represents a transpose of i^{th} training sample that is a vector of n components, x_j represents j^{th} training sample that is another vector of n components, n represents the number of features in the group G. Other embodiments may implement other modifications and variations to block 406. For example, for non-linear SVM training, the kernel data may be obtained in a different way, e.g., the Gaussian RBF kernel:

$$k_{i,j}^{round} = e^{-\|x_i - x_j\|^2 / 2\sigma^2} \tag{3}.$$

[0035] Then, the kernel data logic 220 stores the kernel data in the buffer 221 of the SVM learning machine 22 in block 403. The Lagrange multiplier logic 222 may compute a Lagrange multiplier matrix based upon the kernel data in blocks 408-412 and the weight logic 223 may compute a weight vector based on the Lagrange multiplier matrix in block 414. With these implementations, the first round of SVM training for the training data 110 is completed.

[0036] However, if the kernel data logic 220 or other suitable logics determines that it is not the first round of SVM training for the training data 110 in block 401, then in block 404, the kernel data logic 220 or other suitable logics may input the at least one feature eliminated in a previous round of feature elimination implemented in block 305 of Fig. 3. For example, if it is qth round of SVM training (q>1), then the kernel data logic or other suitable logics may input the at least one feature eliminated in a (q-1)th round of feature elimination (e.g., the pth feature that is eliminated from the group of n features in the (q-1)th round of feature elimination). Then, the kernel data logic 220 may retrieve the kernel data stored in the buffer 221 in a previous round of SVM training (block 405), and update the kernel data based on a part of the training data corresponding to the at least one eliminated feature (block 406). In an embodiment for linear SVM training, the kernel data may be updated by the following equations (4) and (5):

$$K^{round(q)} = \begin{bmatrix} k_{1,1}^{round(q)} \cdots\cdots\cdots k_{1,m}^{round(q)} \\ \cdots\cdots\cdots k_{i,j}^{round(q)} \cdots\cdots\cdots \\ k_{m,1}^{round(q)} \cdots\cdots\cdots k_{m,m}^{round(q)} \end{bmatrix} \tag{4}$$

$$k_{ij}^{round(q)} = k_{ij}^{round(q-1)} - x_{ip}x_{jp} \tag{5}$$

wherein, $k_{ij}^{round(q)}$ represents a component of the kernel data K in qth round of SVM training, $k_{ij}^{round(q-1)}$ represents a component of the kernel data K in a (q-1)th round of SVM training, x_{ip} represents the ith training sample with pth feature that is eliminated in (q-1)th round of feature elimination, x_{jp} represents the jth training sample with pth feature that is eliminated in (q-1)th round of feature elimination.

[0037] Other embodiments may implement other modifications and variations to block 406. For example, for non-linear SVM training, the kernel data may be updated in a different way, e.g., for the Gaussian RBF kernel, a component for the kernel data K in qth round may be updated by

$$k_{ij}^{round(q)} = k_{ij}^{round(q-1)} \times e^{-(x_{ip} - x_{jp})^2/2\sigma^2} \qquad (6).$$

[0038] Then, in block 407, the kernel data logic 220 may replace the kernel data in the buffer 221 with the updated kernel data obtained in block 406. The Lagrange multiplier logic 222 may compute a Lagrange multiplier matrix based on the kernel data in blocks 408-412 and the weight logic 223 may compute a weight vector based on the Lagrange multiplier matrix in block 414. With these implementations, the qth round of SVM training is completed.

[0039] More specifically, in block 408, the Lagrange multiplier logic 222 may initialize a Lagrange multiplier matrix α in each round of SVM training, wherein each component of the α matrix represents a Lagrange multiplier (e.g. α_i) corresponding to a training sample x_i. In an embodiment, the initialization of the Lagrange multiplier matrix may be implemented by setting a predetermined value (e.g., zero) to each component of the Lagrange multiplier matrix.

[0040] Then, in block 409, the Lagrange multiplier logic 222 may determine whether each of the Lagrange multipliers corresponding to each of the training samples (e.g., [α_1, α_2, ..., α_m]) fulfill the Karush-Kuhn-Tucker (KKT) conditions. More specifically, whether each of the Lagrange multipliers fulfills the following five conditions:

1. $\frac{\partial}{\partial w_v} L(w,b,\alpha) = w_v - \sum_{i=1}^{m} \alpha_i y_i x_{iv} \qquad v = 1,\ldots\ldots,n$
2. $\frac{\partial}{\partial b} L(w,b,\alpha) = -\sum_i \alpha_i y_i = 0$
3. $y_i(x_i \cdot w - b) - 1 \geq 0 \qquad i = 1,\ldots\ldots,m$
4. $\alpha_i \geq 0 \qquad \forall i$
5. $\alpha_i(y_i(x_i \cdot w - b) - 1) = 0$

wherein, w_v represents the weight for the v^{th} feature, b represents a bias value, $L(w,b,\alpha)$ represents a Lagrangian with w, b and α as variables:

$$L(w,b,\alpha) = \frac{1}{2}\langle w \cdot w \rangle - \sum_{i=1}^{m} \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1] \qquad (7)$$

[0041] If not all of the Lagrange multipliers fulfill the KKT conditions, the Lagrange multiplier logic 222 may initialize an active set for two Lagrange multipliers in block 410. In an embodiment, the initialization of the active set may be implemented by clearing a data fragment in a memory of the computing system to store the active set. In other embodiments, the active set may be initialized in other ways.

[0042] Then, in block 411, the Lagrange multiplier logic 222 may select two Lagrange multipliers (e.g., α_1 and α_2) as an active set with heuristics, wherein the two Lagrange multiplier violates the KKT conditions with minimum errors (e.g., errors E_1 and E_2 respectively associated with the two Lagrange multipliers α_1 and α_2) under a predetermined constraint. In order to do that, the Lagrange multiplier logic 222 may obtain the errors associated with each of the Lagrange multipliers (e.g., [α_1, α_2,...., α_m]) by utilizing the kernel data stored in the buffer 221. In an embodiment for linear SVM training, the predetermined constraint may comprise $0 \le \alpha_i \le C$ wherein C is a predetermined value, and the error associated with each Lagrange multiplier may be obtained by the following equation and then stored in an error cache:

$$E_j = \left(\sum_{i=1}^{m} \alpha_i y_i k_{ij}^{round(q)} - y_j \right) \qquad j = 1, \ldots\ldots, m \qquad (8)$$

wherein, E_j represents an error associated with a Lagrange multiplier α_j in qth round of SVM training, $k_{ij}^{round(q)}$ may be obtained from the kernel data stored in the buffer 221. Other embodiments may implement other modifications and variations to block 411. For example, the active set may comprise the number of Lagrange multipliers other than two.

[0045] Then, in block 412, the Lagrange multiplier logic 222 may update the Lagrange multipliers in the active set by utilizing the kernel data K stored in the buffer 221. In an embodiment that the SVM learning machine is a linear learning machine and the active set may comprise two Lagrange multipliers (e.g., α_1 and α_2), the Lagrange multiplers may be updated with the following equations:

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_2 - E_1)}{\eta}, \;\; \eta \equiv 2k_{12} - k_{11} - k_{22}, \; E_j = \left(\sum_{i=1}^{m} \alpha_i y_i k_{ij}^{round(q)} - y_j \right) - y_j \qquad (9)$$

$$\alpha_2^{new,clipped} = \begin{cases} H & if \quad \alpha_2^{new} \geq H \\ \alpha_2^{new} & if \quad L < \alpha_2^{new} < H \\ L & if \quad \alpha_2^{new} \leq L \end{cases} \tag{10}$$

$$L = \max(0, \alpha_2 - \alpha_1), \qquad H = \min(C, C + \alpha_2 - \alpha_1) \tag{11}$$

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new,clipped}), s = y_1 y_2 \tag{12}$$

[0044] However, other embodiments may implement other modifications and variations to block 412.

[0045] Then, in block 413, the Lagrange multiplier logic 222 may update the error cache by computing the errors associated with the updated Lagrange multipliers in the active set with the equation (8).

[0046] Then, the Lagrange multiplier logic 222 may continue to update other Lagrange multipliers in the Lagrange multiplier matrix in blocks 408-413, until all of the Lagrange multipliers in the matrix fulfill KKT conditions.

[0047] Then, the weight logic 223 may compute the weight vector ($\vec{\omega}$) based on the Lagrange multipliers obtained in blocks 408-413, wherein each component of the vector corresponds to each of the feature. In an embodiment for linear SVM training, weight for each feature may be obtained with the following equation:

$$w_k = \sum_{i=1}^{m} \alpha_i y_i x_{ik} \tag{13}$$

wherein, w_k represents a weight for k^{th} feature, m represent the number of the training samples, x_{ik} represents the training samples corresponding to the

k^{th} feature. However, other embodiments may implement other modifications and variations to block 414.

[0048] Although the present invention has been described in conjunction with certain embodiments, it shall be understood that modifications and variations may be resorted to without departing from the spirit and scope of the invention as those skilled in the art readily understand. Such modifications and variations are considered to be within the scope of the invention and the appended claims.